Bidirectional Contextual Grammars

Jiří Techet *
techet@fit.vutbr.cz

Abstract: The present paper introduces and discusses bidirectional contextual grammars as a straightforward generalization of externally generating contextual grammars without choice. In essence, besides ordinary derivation steps, the bidirectional contextual grammars can also make reduction steps, which shorten the rewritten strings. This paper demonstrates that these grammars characterize the family of recursively enumerable languages. In fact, this characterization holds even in terms of one-turn bidirectional contextual grammars, which can change derivations steps to reduction steps during the generation process no more than once.

Keywords: contextual grammars, bidirectional grammars, generative power, recursively enumerble languages

1 Introduction

Over its history, the language theory has always paid a special attention to the Marcus contextual grammars because these grammars fulfill a significant role in the generation of both natural and formal languages (see [7], [8], chapter 5 and 6 in Volume II of [14]). It thus come as no surprise that the language theory has discussed a large variety of these grammars (see [12]). This paper contributes to this trend by investigating another variant of these grammars whose introduction is inspired by two grammatically oriented studies in the formal language theory. First, more than three decades ago, this theory used grammars with special end markers during the generation of languages (see page 99 in [15]). Second, about two decades ago, the language theory introduced various bidirectional grammars that both derive and reduce strings during their generation process (see [1], [2], [4], [5], [11]). These two studies have given rise to the variant of contextual grammars discuss in this paper.

More specifically, this paper introduces *bidirectional contextual grammars* as a straightforward generalization of the externally generating contextual grammars without choice (see page 240 in Volume II of [14]). A bidirectional contextual grammar, G, is based on derivation and reduction rules of the form (x, y), where x and y are strings. From a string z, G makes a derivation step by using a derivation rule, (u, v), like in any externally generating contextual grammars—that is, it changes z to uzv by using this derivation rule. In addition, however, by using a reduction rule, (t, w), from tzw, G makes a reduction step so it changes tzw to z. If G can make a computation from G's axiom, s, to $\$z\$$, where $\$$ is a special bounding symbol, z is in the language defined by G. Two consecutive computational steps G makes are called a *turn* if one is a reduction step and the other represents a derivation step. Let i be a non-negative integer. G is an *i-turn* bidirectional contextual grammar if G makes no more than i turns during every generation of string from its language.

* Department of Information Systems, Faculty of Information Technology, Brno University of Technology, Bozetechova 2, Brno 61266, Czech Republic

2 Preliminaries

We assume that the reader is familiar with the language theory (see [9], [13], [14], [15]). For an alphabet, V, $\mathrm{card}(V)$ denotes the cardinality of V. V^* represents the free monoid generated by V under the operation of concatenation. The unit of V^* is denoted by ε. Set $V^+ = V^* - \{\varepsilon\}$. For $w \in V^*$, $|w|$ and $\mathrm{alph}(w)$ denote the length of w and the set of symbols occurring in w, respectively. For $L \subseteq V^*$, $\mathrm{alph}(L) = \{a : a \in \mathrm{alph}(w), w \in L\}$. For $w \in V^*$ and for $a \in V$, $\mathrm{occur}(w, a)$ denotes the number of occurrences of a in w. For $w \in V^*$, $\mathrm{prefix}(w)$ and $\mathrm{suffix}(w)$ denote the set of all w's prefixes and suffixes, respectively. For $(w_1, \dots, w_n) \in V_1^* \times \dots \times V_n^*$, where $V_1, \dots, V_n$ are finite alphabets, $\mathrm{concat}((w_1, \dots, w_n)) = w_1 \dots w_n$.

A *queue grammar* (see [6]) is a sixtuple,

$$Q = (V, T, W, F, s, P),$$

where V and W are alphabets satisfying $T \subseteq V$, $F \subseteq W$, $s \in (V - T)(W - F)$, and

$$P \subseteq (V \times (W - F)) \times (V^* \times W)$$

is a finite relation whose elements are called productions. For every $a \in V$, there exists a production $(a, b, x, c) \in P$. If $u, v \in V^*W$ such that $u = arb$, $v = rxc$, $a \in V$, $r, x \in V^*$, $b, c \in W$, and $(a, b, x, c) \in P$, then

$$u \Rightarrow v \; [(a, b, x, c)]$$

in G or, simply, $u \Rightarrow v$. In the standard manner, extend $\Rightarrow$ to $\Rightarrow^n$, where $n \geq 0$; then, based on $\Rightarrow^n$, define $\Rightarrow^+$ and $\Rightarrow^*$. The language of Q, $L(Q)$, is defined as

$$L(Q) = \{w \in T^* : s \Rightarrow^* wf \text{ where } f \in F\}.$$

A *left-extended queue grammar* is a sixtuple,

$$Q = (V, T, W, F, s, P),$$

where V, T, W, F, s, and P have the same meaning as in a queue grammar; in addition, assume that $\# \notin V \cup W$. If $u, v \in V^*\{\#\}V^*W$ so that $u = w\#arb$, $v = wa\#rxc$, $a \in V$, $r, x, w \in V^*$, $b, c \in W$, and $(a, b, x, c) \in P$, then

$$u \Rightarrow v \; [(a, b, x, c)]$$

in G or, simply, $u \Rightarrow v$. In the standard manner, extend $\Rightarrow$ to $\Rightarrow^n$, where $n \geq 0$; then, based on $\Rightarrow^n$, define $\Rightarrow^+$ and $\Rightarrow^*$. The language of Q, $L(Q)$, is defined as

$$L(Q) = \{v \in T^* : \#s \Rightarrow^* w\#vf \text{ for some } w \in V^* \text{ and } f \in V^*\}.$$

3 Definitions

A *bidirectional contextual grammar* is a triple

$$G = (T \cup \{\$\}, P_d \cup P_r, S),$$

where T is an alphabet, $\$$ a special symbol, $\$ \notin T$,

$$P \subseteq (T \cup \{\$\})^* \times (T \cup \{\$\})^*,$$

$P = P_d \cup P_r$, and S is a finite language over T. For every $x \in (T \cup \{\$\})^*$ and $(u, v) \in P_d$, write

$$x \ {}_d\!\Rightarrow uxv,$$

and for every $(u, v) \in P_r$, write

$$uxv \ {}_r\!\Rightarrow x;$$

intuitively, d and r stand for a direct *d*erivation and a direct *r*eduction, respectively. To express that G makes $x \ {}_d\!\Rightarrow uxv$ according to (u, v), write $x \ {}_d\!\Rightarrow uxv \ [(u, v)]$; $uxv \ {}_r\!\Rightarrow x \ [(u, v)]$ has an analogical meaning in terms of ${}_r\!\Rightarrow$. Let $y, x \in (T \cup \{\$\})^*$. We say that G makes a *direct computation of x from y*, symbolically written as

$$y \Rightarrow x,$$

if either $y \ {}_d\!\Rightarrow x$ or $y \ {}_r\!\Rightarrow x$ in G. In the standard manner, extend $\Rightarrow$ to $\Rightarrow^m$, where $m \geq 0$; then, based on $\Rightarrow^m$, define $\Rightarrow^+$ and $\Rightarrow^*$. The *\$-bounded language generated by G*, ${}_\$L(G)$, is defined as

$${}_\$L(G) = \{z \,:\, s \Rightarrow^* \$z\$ \text{ in } G, z \in T^*, s \in S\}.$$

A computation of the form $s \Rightarrow^* \$z\$$ in G, where $z \in T^*$ and $s \in S$, is said to be *successful*. Any two-step computation, $y \Rightarrow^2 x$, where $y, x \in (T \cup \{\$\})^*$, represents a *turn* if $y \Rightarrow^2 x$ is of the form

$$y \ {}_d\!\Rightarrow z \ {}_r\!\Rightarrow x$$

or

$$y \ {}_r\!\Rightarrow z \ {}_d\!\Rightarrow x,$$

for some $z \in (T \cup \{\$\})^*$; less formally, the two-step computation $y \Rightarrow^2 x$ consists of one direct derivation and one direct reduction. G is *i-turn* if any successful computation in G contains no more than i turns.

4 Results

This section demonstrates that every recursively enumerable language is defined by a one-turn bidirectional contextual grammar.

Lemma 1. *For every recursively enumerable language, L, there is a left-extended queue grammar, G, such that $L = L(G)$.*

Proof. See Lemma 1 in [6].

□

Lemma 2. *Let Q' be a left-extended queue grammar. Then, there exists a left-extended queue grammar, $Q = (V, T, W, F, s, R)$, such that $L(Q') = L(Q)$, $W = X \cup Y \cup \{1\}$, where $X, Y, \{1\}$ are pairwise disjoint, and every $(a, b, x, c) \in R$ satisfies either $a \in V - T$, $b \in X$, $x \in (V - T)^*$, $c \in X \cup \{1\}$ or $a \in V - T$, $b \in Y \cup \{1\}$, $x \in T^*$, $c \in Y$.*

Proof. See Lemma 1 in [10].

□

Consider the left-extended queue grammar, $Q = (V, T, W, F, s, R)$, from Lemma 2. Its properties imply that Q generates every word in $L(Q)$ so that it passes through state 1. Before it enters 1, it generates only words over $V - T$; after entering 1, it generates only words over T. In greater detail, the next corollary expresses this property, which fulfills a crucial role in the proof of Theorem 4.

Corollary 3. *Q constructed in the proof of Lemma 2 generates every $h \in L(Q)$ in this way*

$$\begin{array}{lll} & \#a_0q_0 & \\ \Rightarrow & a_0\#x_0q_1 & [(a_0, q_0, z_0, q_1)] \\ \Rightarrow & a_0a_1\#x_1q_2 & [(a_1, q_1, z_1, q_2)] \\ & \vdots & \\ \Rightarrow & a_0a_1 \dots a_k\#x_kq_{k+1} & [(a_k, q_k, z_k, q_{k+1})] \\ \Rightarrow & a_0a_1 \dots a_ka_{k+1}\#x_{k+1}y_1q_{k+2} & [(a_{k+1}, q_{k+1}, y_1, q_{k+2})] \\ & \vdots & \\ \Rightarrow & a_0a_1 \dots a_{k+m-1}\#x_{k+m-1}y_1 \dots y_{m-1}q_{k+m} & [(a_{k+m-1}, q_{k+m-1}, y_{m-1}, q_{k+m})] \\ \Rightarrow & a_0a_1 \dots a_{k+m-1}a_{k+m}\#y_1 \dots y_mq_{k+m+1} & [(a_{k+m}, q_{k+m}, y_m, q_{k+m+1})], \end{array}$$

where $k, m \geq 1$, $a_i \in V-T$ for $i = 0, \dots, k+m$, $x_j \in (V-T)^*$ for $j = 1, \dots, k+m$, $s = a_0q_0$, $a_jx_j = x_{j-1}z_j$ for $j = 1, \dots, k$, $a_1 \dots a_kx_k = z_0 \dots z_k$, $a_{k+1} \dots a_{k+m} = x_k$, $q_0, q_1, \dots, q_k \in X$, $q_{k+1} = 1, q_{k+2}, \dots, q_{k+m} \in Y$ and $q_{k+m+1} \in F$, $z_0, \dots, z_k \in (V-T)^*$, $y_1, \dots, y_m \in T^*$, $h = y_1y_2 \dots y_{m-1}y_m$. □

Theorem 4. *Let L be a recursively enumerable language. Then, there exists a one-turn bidirectional contextual grammar, G, such that $L = {}_{\$}L(G)$.*

Proof. Let L be a recursively enumerable language. Let $Q = (V, T, W, F, s, R)$ be a left-extended queue grammar such that $L(Q) = L$ and Q satisfies the properties described in Lemma 2 and Corollary 3. Select a symbol, $o \in T$. Define the injection, α, from R to $\{o\}^+$ so that α is an injective homomorphism when its domain is extended to R^*. Further, define the binary relation, f, over V so that $f(\varepsilon) = \varepsilon$ and

$$f(a) = \{\alpha((a, b, c_1 \dots c_n, d)) \,:\, (a, b, c_1 \dots c_n, d) \in R\}$$

for all $a \in V$. Similarly, define the binary relation, g, over W so that

$$g(b) = \{\alpha((a, b, c_1 \dots c_n, d)) \,:\, (a, b, c_1 \dots c_n, d) \in R\}$$

for all $b \in W$. In the standard manner, extend the domain of f and g to V^* and W^*, respectively. Define the bidirectional contextual grammar,

$$G = (T \cup \{\$\}, P_d \cup P_r, S),$$

with

$$S = \{\, c_1 \dots c_n\$\alpha((a, b, c_1 \dots c_n, d)) \,:\, (a, b, c_1 \dots c_n, d) \in R, \\ c_1, \dots, c_n \in T \text{ for some } n \geq 0, d \in F\}$$

and P_d, P_r constructed as follows:

1. For every $(a, b, c_1 \dots c_n, d) \in R$, $c_1, \dots, c_n \in T$, for some $n \geq 0$, $d \in (W-F)$, $\overline{d} \in g(d)$, add
   $(c_1 \dots c_n, \$\$\overline{d}\$\$\alpha((a, b, c_1 \dots c_n, d)))$ to P_d;
2. For every $(a, b, c_1 \dots c_n, d) \in R$, $c_1, \dots, c_n \in (V-T)$, for some $n \geq 0$, $d \in (W-F)$, $\overline{c}_1 \in f(c_1), \dots, \overline{c}_n \in f(c_n)$, $\overline{d} \in g(d)$, add
   $(\overline{c}_1\$\overline{c}_2\$ \dots \overline{c}_n\$, \$\$\overline{d}\$\$\alpha((a, b, c_1 \dots c_n, d)))$ to P_d;
3. For every $\overline{a}_0 \in f(a_0)$, $\overline{q}_0 \in g(q_0)$ such that $s = a_0q_0$, add
   $(\$\overline{a}_0\$, \$\$\overline{q}_0\$\$)$ to P_d;
4. For every $r \in R$, add
   $(\$\alpha(r), \alpha(r)\$\$\alpha(r)\$\$)$ to P_r.

Denote the set of productions introduced in step i of the construction by ${}_iP_d$, for $1 \leq i \leq 3$.

Basic Idea:

Each simulation consists of two phases. In the first phase, the simulation of Q's derivation is performed nondeterministically, and in the second phase, this simulation is verified. Next, we sketch both phases in greater detail.

Simulation phase. G performs the simulation of Q in reverse. That is, the last derivation step in Q is simulated first in G, and the first step in Q is simulated last in G. In general, G keeps the binary code of the string over V that Q generates as a prefix of the current sentential form while keeping the binary code of states as its suffix. By a string from S, the Q's production of the form $p_0 : (a, b, c_1 \dots c_n, d)$, $d \in F$ is simulated; it places $c_1 \dots c_n$ as the prefix and the p_0's code as the suffix of the sentential form. Then, productions $p_1 : (a, b, c_1 \dots c_n, d)$, $c_1, \dots, c_n \in T$, $d \in (W - F)$ are simulated by productions from ${}_1P_d$. They place $c_1 \dots c_n$ as the prefix and the codes of d and p_1 as the suffix of the sentential form. The suffix codes of the sentential form are always separated by \$\$. Productions from ${}_2P_d$ simulate productions $p_2 : (a, b, c_1 \dots c_n, d)$, $c_1, \dots, c_n \in (V - T)$; they place codes of $c_1, \dots, c_n$ as the prefix and, again, d's and p_2's code as the suffix. The prefix codes are always separated by \$. Finally, by productions from ${}_3P_d$, the axiom $s = a_0q_0$ from Q is simulated.

Verification phase. During every step of the verification phase, G makes sure that the two suffix binary codes and the prefix binary code correspond to the same production in Q. If they do, all these three codes are removed from the sentential form. In this way, step by step, G verifies that the previously made simulation phase was performed properly.

Rigorous Proof:

Lemma 5. *G generates every $w \in {}_\$L(G)$ in the following way:*

$$\begin{aligned} s &\Rightarrow^+ v\ [\rho] \\ &{}_r\Rightarrow^+ w\ [\tau], \end{aligned} \tag{1}$$

where $s \in S$, $v \in \{\$\}p_n\{\$\}p_{n-1} \dots \{\$\}p_1\{\$\}w\{\$\}p_1\{\$\}\{\$\}p_1\{\$\}\{\$\} \dots p_{n-1}\{\$\}\{\$\}p_{n-1}\{\$\}\{\$\}p_n\{\$\}\{\$\}p_n\{\$\}\{\$\}$, $n \geq 1$, where $\alpha^{-1}(p_1) \in R, \dots, \alpha^{-1}(p_n) \in R$ and $w \in T^*$. Sequences ρ and τ denote productions from ${}_1P_d \cup {}_2P_d \cup {}_3P_d \cup P_r$ and P_r, respectively.

Proof. Observe that every $p_d \in {}_1P_d \cup {}_2P_d \cup {}_3P_d$ satisfies $\text{occur}(\text{concat}(p_d), \$) \geq 4$. By the definition of ${}_\$L(G)$, however, precisely two \$s occur in the generated sentence. Therefore, no production from ${}_1P_d \cup {}_2P_d \cup {}_3P_d$ is applied in the very last computation step. For the same reason, $s \notin {}_\$L(G)$, for any $s \in S$, because $occur(s, \$) = 1$ for every $s \in S$. Therefore,

$$\begin{aligned} s &\Rightarrow^+ v\ [\rho] \\ &{}_r\Rightarrow^+ w\ [\tau]. \end{aligned}$$

Observe that an application of every production from P_d removes three identical codes $\alpha(r)$, $r \in R$. More specifically, one prefix code and two subsequent suffix codes of the sentential form are removed. Furthermore, notice that the prefix codes and the suffix codes of the sentential form have to be separated by \$ and \$\$, respectively. Next, we consider the following four forms of v and demonstrate that G can generate a member of ${}_\$L(G)$ from none of them.

1. Let $v \in \{\$\}p_{n-1} \dots \{\$\}p_1\{\$\}w\{\$\}p_1\{\$\}\{\$\}p_1\{\$\}\{\$\} \dots p_{n-1}\{\$\}\{\$\}p_{n-1}\{\$\}\{\$\}p_n\{\$\}\{\$\}p_n\{\$\}\{\$\}$, $n \geq 1$, where $\alpha^{-1}(p_1) \in R, \dots, \alpha^{-1}(p_n) \in R$ (notice that $\{\$\}p_n \notin \text{prefix}(v)$). In the case that $p_{i-1} = p_i$ for all $1 \leq i \leq n$ and $w = p_1$, the reduction phase can be preformed. After all applicable reducing productions are used, the resulting sentence consists only from \$, which is not in ${}_\$L(G)$.

2. Let $v \in \{\$\}p_{n-2}\ldots\{\$\}p_1\{\$\}w\{\$\}p_1\{\$\}\{\$\}p_1\{\$\}\{\$\}\ldots p_{n-1}\{\$\}\{\$\}p_{n-1}\{\$\}\{\$\}p_n\{\$\}$ $\{\$\}p_n\{\$\}\{\$\}$ (notice that $\{\$\}p_n\{\$\}p_{n-1} \notin \text{prefix}(v)$). In the case that $p_{i-2} = p_i$ for all $1 \leq i \leq n$, $p_1 = p_2$ and $w = p_3$, the reduction phase can be preformed. The resulting sentence satisfies $\{\$\}\{\$\}p_1\{\$\}\{\$\}$, which is not in ${}_{\$}L(G)$ either.
3. Let $v \in \{\$\}p_n\{\$\}p_{n-1}\ldots\{\$\}p_1\{\$\}w\{\$\}p_1\{\$\}\{\$\}p_1\{\$\}\{\$\}\ldots p_{n-1}\{\$\}\{\$\}p_{n-1}\{\$\}\{\$\}$ (notice that $p_n\{\$\}\{\$\}p_n\{\$\}\{\$\} \notin \text{suffix}(v)$). In the case that $p_{i-1} = p_i$ for all $1 \leq i \leq n$, the reduction phase can be preformed. In this case, however, the reduction ends when the sentential form satisfies $\{\$\}p_1\{\$\}w\{\$\}$, so the computation ends unsuccessfully as well.
4. All other cases, where the number of prefix codes and suffix codes of v does not match, lead to an unsuccessful computation.

Hence, to generate a string $w \in {}_{\$}L(G)$, the sentential form v has to have the form $v \in \{\$\}p_n\{\$\}p_{n-1}\ldots\{\$\}p_1\{\$\}w\{\$\}p_1\{\$\}\{\$\}p_1\{\$\}\{\$\}\ldots p_{n-1}\{\$\}\{\$\}p_{n-1}\{\$\}\{\$\}p_n\{\$\}\{\$\}p_n$ $\{\$\}\{\$\}$. Therefore, this lemma holds.

Lemma 6. *Every $s \Rightarrow^+ v$ from (1) described in Lemma 5 can be expressed in greater detail as*

$$\begin{aligned} s \;{}_d\!\Rightarrow^* &\; u \; [\xi] \\ {}_d\!\Rightarrow &\; v \; [p_n], \end{aligned}$$

where $u \in p_{n-1}\{\$\}p_{n-2}\ldots\{\$\}p_1\{\$\}w\{\$\}p_1\{\$\}\{\$\}p_1\{\$\}\{\$\}\ldots p_{n-1}\{\$\}\{\$\}p_{n-1}\{\$\}\{\$\}p_n$, $n \geq 1$ with $\alpha^{-1}(p_1) \in R, \ldots, \alpha^{-1}(p_n) \in R$. Production $p_n \in {}_3P_d$ has the form $(\$\alpha(r_n)\$, \$\$\alpha(r_n)\$\$)$, $r_n \in R$, ξ represents a sequence of productions from ${}_1P_d \cup {}_2P_d$.

Proof. First, observe that every $s \in S$ satisfies $\$ \notin \text{suffix}(s)$, and every production $p_d \in {}_1P_d \cup {}_2P_d$ satisfies $\$ \notin \text{suffix}(\text{concat}(p_d))$. As every $p_r \in P_r$ satisfies $\$\$ \in \text{suffix}(\text{concat}(p_r))$, p_r cannot be used after an application of production from ${}_1P_d \cup {}_2P_d$. As $\$\$ \in \text{suffix}(\text{concat}(p_3))$ and $\$ \in \text{prefix}(\text{concat}(p_3))$ for all $p_3 \in {}_3P_d$, after an application of p_3 productions from P_r can be used. Therefore, the computation can be expressed as

$$\begin{aligned} s \;{}_d\!\Rightarrow^* &\; u_1 \; [\xi_1] \\ {}_d\!\Rightarrow &\; u_2 \; [p_3] \\ \Rightarrow^* &\; v \;\; [\xi_2], \end{aligned}$$

where ξ_1 and ξ_2 are sequences of productions from ${}_1P_d \cup {}_2P_d$ and ${}_1P_d \cup {}_2P_d \cup {}_3P_d \cup P_r$, respectively. Next, we show that $u_2 \Rightarrow^0 v$ and $\xi_1 = \xi$.

Now, we demonstrate that after any application of a production from P_r, the sentential form w satisfies $\$ \in \text{suffix}(w)$. In what follows, $\alpha^{-1}(p_0) \in R, \ldots, \alpha^{-1}(p_n) \in R, \alpha^{-1}(p'_1) \in R, \ldots, \alpha^{-1}(p'_{n+1}) \in R$ for some $n \geq 0$. The generation starts from $s \in S$, which is of the form $T^*\{\$\}p_0$. Productions from ${}_1P_d \cup {}_2P_d$ have the form $((T \cup \{\$\})^*, \{\$\}\{\$\}p'_i\{\$\}\{\$\}p_i)$, $1 \leq i \leq n$; after their n applications followed by one application of a production from ${}_3P_d$ of the form $((T \cup \{\$\})^*, \{\$\}\{\$\}p'_{n+1}\{\$\}\{\$\})$, we obtain $(T \cup \{\$\})^* T^*\{\$\}p_0\{\$\}\{\$\}p'_1\{\$\}\{\$\}p_1\ldots\{\$\}\{\$\}p'_n\{\$\}$ $\{\$\}p_n\{\$\}\{\$\}p'_{n+1}\{\$\}\{\$\}$. If $p_i = p'_{i+1}$, $1 \leq i \leq n$ and the prefix of the sentential form satisfies some other requirements, productions from P_r can start performing reduction so that they erase $p_i\{\$\}\{\$\}p'_{i+1}\{\$\}\{\$\}$ from the suffix and some other symbols from the prefix of the sentential form. Finally, when we get by this reduction a sentential form of the form $(T \cup \{\$\})^* T^*\{\$\}$ no other productions from P_r can be used. Observe that after any application of a production from P_r, the sentential form w satisfies $\$ \in \text{suffix}(w)$.

The sentential form w satisfies $\$ \in \text{suffix}(w)$ after an application of a production from P_r, and every production $p_d \in {}_1P_d \cup {}_2P_d \cup {}_3P_d$ has the form $((T \cup \{\$\})^*, \{\$\}\{\$\}(T \cup \{\$\})^*)$.

Therefore, by the derivation $w\ {}_d\!\Rightarrow w_1\ [p_d]$, we get $\$\$\$ \in \text{suffix}(w_1)$ so the computation is blocked. For this reason, only productions from P_r can be used after the application of production from ${}_3P_d$. To successfully reduce the sentential form, u, observe that u has to follow the form described in the formulation of this lemma.

Putting together the previous two lemmas, we see that every successful computation is of the form

$$\begin{aligned} S\ &{}_d\!\Rightarrow^* u\ [\xi] \\ &{}_d\!\Rightarrow v\ [p_3] \\ &{}_r\!\Rightarrow^* w\ [\eta], \end{aligned}$$

where $p_3 \in {}_3P_d$, ξ and η are sequences of productions from ${}_1P_d \cup {}_2P_d$ and P_r, respectively.

The bidirectional contextual grammar, G, simulates the queue grammar, Q, in reverse by a string from S and productions ${}_1P_d \cup {}_2P_d \cup {}_3P_d$. Productions from P_r perform the verification of this simulation in the same order as Q makes its derivation steps. The succession of states is checked in the suffix of the sentential form, the queue is simulated in the prefix. As we expect Q to satisfy Lemma 2 and Corollary 3 and the succession of states is checked, we can express the computation as a whole in a more detailed way:

$$\begin{aligned} S\ &{}_d\!\Rightarrow^* y\ [\psi] \\ &{}_d\!\Rightarrow^* u\ [\zeta] \\ &{}_d\!\Rightarrow v\ [p_3] \\ &{}_r\!\Rightarrow^* w\ [\eta], \end{aligned}$$

where ψ and ζ denote sequences of productions from ${}_1P_d$ and ${}_2P_d$, respectively. Examine this computation and the productions to see that $w \in {}_\$L(G)$ if and only if $w \in L(Q)$.

□

5 Conclusion

As its main result, this paper proves that the bidirectional one-turn contextual grammars characterize the family of recursively enumerable languages. This result is of some interest because externally generating contextual grammars without choice define only the family of minimal linear languages (see Lemma 2.9 on page 247 in Volume II of [14]). In fact, every recursively enumerable language is defined by a one-turn bidirectional contextual grammar.

Bibliography

1. D. E. Appelt. Bidirectional grammars and the design of natural language generation systems. In *Proceedings of Third Conference on Theoretical Issues in Natural Language Processing (TINLAP-3)*, pages 185–191, Las Cruces, New Mexico, 1987.
2. P. R. J. Asveld and J. A. Hogendorp. On the generating power of regularly controlled bidirectional grammars. *International Journal of Computer Mathematics*, 40:75–91, 1991.
3. H. Fernau and M. Holzer. Bidirectional cooperating distributed frammar systems. Technical Report WSI-96-01, Wilhelm Schickard-Institut fur Informatik, Germany, 1996.
4. J. A. Hogendorp. Controlled bidirectional grammars. *International Journal of Computer Mathematics*, 27:159–180, 1989.
5. J. A. Hogendorp. Time-bounded controlled bidirectional grammars. *International Journal of Computer Mathematics*, 35:93–115, 1990.

6. H. C. M. Kleijn and G. Rozenberg. On the generative power of regular pattern grammars. *Acta Informatica*, 20:391–411, 1983.
7. S. Marcus, editor. *Algebraic Linguistics. Analytical Models.* Academic Press, New York, 1967.
8. S. Marcus. Contextual grammars. *Rev. Roum. Math. Pures Appl.*, 14:1525–1534, 1969.
9. A. Meduna. *Automata and Languages: Theory and Applications.* Springer-Verlag, London, 2000.
10. A. Meduna. Simultaneously one-turn two-pushdown automata. *International Journal of Computer Mathematics*, 80:679–687, 2003.
11. A. Meduna. Two-way metalinear pc grammar systems and their descriptional complexity. *Acta Cybernetica*, 16:385–397, 2004.
12. G. Paun. *Marcus contextual grammars.* Kluwer Academic Publishers, London, 1997.
13. G. E. Revesz. *Introduction to Formal Language Theory.* McGraw-Hill, New York, 1983.
14. G. Rozenberg and A. Salomaa, editors. *Handbook of Formal Languages*, volume 1 through 3. Springer-Verlag, Berlin, 1997.
15. A. Salomaa. *Formal Languages.* Academic Press, New York, 1973.